Swarm Dispersion via Potential Fields, Leader Election, and Counting Hops

Anuraag Pakanati and Maria Gini

Department of Computer Science and Engineering, University of Minnesota
pakanati@msu.edu,gini@cs.umn.edu

Abstract. We examine how robots using only laser range data and the ability to communicate within a certain radius may disperse in a complex unknown environment. We demonstrate an algorithm which attempts to maximize coverage while minimizing loss of connectivity during deployment. We demonstrate using Player/Stage that this algorithm can reliably achieve significant coverage over a variety of environments.

1 Introduction

Consider a domain such as a cave or a disaster area in which the layout is unknown ahead of time and where we need to create a wireless network in order to facilitate the deployment of men and material, or gather and transmit sensor information. Automated methods to accomplish this are highly desirable since not only they avoid placing humans in harm's way, but they allow human resources to be focused, either elsewhere or more precisely with information gathered by such a network.

Robot swarm dispersion refers to a broad class of problems in which a collection of robots starts out in a relatively compact space and must spread out in order to cover as much area as possible while preserving connectivity of the network. Different applications typically have different requirements. For example, robustness to node failure may be important in a tracking application such as radar, but less important in other applications. Metrics for the effectiveness of dispersion involve evaluating the amount of coverage achieved at a certain time, the rate of coverage increase over time, the time needed to deploy the network, robustness to node failure, and degree of overcoverage, among others.

Our robots lack GPS, maps, compass, or any form of absolute localization, but do have access to laser range-finders and wireless communication. We use a model-free approach where robots achieve coverage in a distributed fashion and do not attempt to build a map of the environment. We build upon artificial potential fields [1] and use connectivity information in order to preserve network connectivity while achieving dispersion. Our approach is empirical due to the extreme difficulty of providing theoretical guarantees on the performance of such systems. We present two major contributions:

1. a novel algorithm to enable dispersion of a group of minimally equipped robots in an unknown environment. The algorithm uses leader-election, hop

counting, and potential fields to maximize coverage while maintaining the robots within communication range of each other.
2. extensive experimental results obtained in simulation in a variety of environments. The results show the effectiveness of the algorithm in achieving a significant coverage while maintaining connectivity.

In Section 2, we discuss relevant background literature. We present our algorithm in Section 3, followed in Section 4 by experimental results, where we measure the performance of our algorithm in four simulated environments. Finally, in Section 5 we wrap up with conclusions and discussion of future work.

2 Related Work

A common way of achieving robot dispersion is to use artificial potential fields [1] to repel robots away from obstacles and attract them to their goal. There are many variations in the way artificial potential fields are used, but all approaches assume that the relative locations (distance and bearing) of obstacles is available to each robot through its sensors or communication system.

Dispersion in an unknown environment was tackled in [2] using various behaviors [3] to achieve dispersion but without regard for network connectivity. In [4] robots equipped with 360° laser range-finders use potential fields and a viscous friction term in order to assure that the network converges. This is similar to our work, since we also use 360° lasers and potential fields, but we use communication and connectivity to prevent disconnection of the network. In [5] mobile nodes are treated as charged particles with two forces, one repulsive that tries to maximize coverage, the other attractive that attempts to maintain for each node not on the boundary at least one neighbor in every θ sector of communication range. This requires each robot to know the relative position of its neighbors. There are only convex obstacles to limit the ability of the nodes to move, so the approach does not work for deploying a network in a building.

To maintain connectivity in [6] robots are deployed sequentially using various heuristics to select the place for the next robot. Because of the sequential deployment the process is very slow. The LRV algorithm [7] uses a single robot carrying around network nodes in an unexplored area and deploying them. The deployment is again sequential, one sensor at the time. A dispersion algorithm that uses virtual pheromones similar to those in [8] enables small robots to explore [9]. The communication system provides range and bearing to any robot in the communication range, and the space is open compared to the robot size.

Our work most closely resembles the clique algorithm [10], which uses wireless intensity signals to prevent robots from running out of reach from each other. Robots share connectivity information with each other so that they can all agree on which robots should act as “sentries” and not move, allowing the other robots to move. Additionally, we explored some of the Neighbor-Every-Theta connectivity ideas in relation to potential fields [4,5], although we assume different capabilities which alters how those ideas are applied. Lastly, we use leader-election and hop-counting to help guarantee connectivity.

3 Dispersion Algorithm

Our algorithm combines artificial potential fields and behaviors with selecting a leader of the swarm, counting hops from the leader, and sending alarms to prevent disconnection from the network.

3.1 Artificial Potential Fields

In our artificial potential fields we use three forces. The first, and largest, is a *radial force* $\mathbf{F}_R$ whose magnitude is determined by distance to obstacles. We use a combination of linear and exponential magnitudes to determine the force, as shown in (1). Here, $\mathbf{p}$ is a laser measurement corresponding to a point mass (obstacle) along a vector, d($\mathbf{p}$) is the distance of that point mass and m($\mathbf{p}$) is the magnitude of the resulting force exerted on the robot by that point, I is an indicator function which evaluates to 1 if its condition is true and 0 otherwise.

$$\mathbf{F}_R = \sum_{\mathbf{p}\in P} -m(\mathbf{p})\mathbf{p}, \text{ where } m(\mathbf{p}) = 4/d(\mathbf{p}) + I_{d(\mathbf{p})\leq 1}e^{-d(\mathbf{p})}) \tag{1}$$

The second force is an *open force* $\mathbf{F}_O$ that pulls a robot towards empty regions. It is generated by a two pass algorithm which first identifies in the range-finder data consecutive readings above a given threshold and creates an attractive force towards the largest opening found. Let $\mathbf{O}$ be the collection of all openings and $\mathbf{O}_C$ the vector pointing outwards towards the center of the open area, then:

$$\mathbf{F}_O = \max_{O\in\mathbf{O}} |\mathbf{O}| \times \mathbf{O}_C \tag{2}$$

Finally, a *tangential force* is generated from each pair $\mathbf{p}$ and $\mathbf{p}'$ of successive point readings. There are two candidates with opposite directions. The ambiguity is broken by using the one with the smaller angle relative to $\mathbf{F}_R$. The magnitude of $\mathbf{f}_T(\mathbf{p},\mathbf{p}')$ is determined by the closer of the two points, and is smaller than $\mathbf{F}_R$. The total tangent force $\mathbf{F}_T$ is simply this force calculated over every pair of successive readings, $\mathbf{p}$ and $\mathbf{p}'$:

$$\mathbf{F}_T = \sum_{\{\mathbf{p},\mathbf{p}'\}\in P} \mathbf{f}_T(\mathbf{p},\mathbf{p}') \tag{3}$$

where, as illustrated in Figure 1:

$$\mathbf{f}_T(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}-\mathbf{B}}{\|\mathbf{A}-\mathbf{B}\|} \cdot \mathrm{sign}((\mathbf{A}-\mathbf{B}) \circ \mathbf{F}_R \cdot \frac{\max(m(\mathbf{A}), m(\mathbf{B}))}{4} \tag{4}$$

Potential fields offer no innate support for preserving connectivity between robots. Potential fields also lend themselves to a few problems. The first issue we discovered in our experiments was that the runs were very sensitive to minor random fluctuations. Despite having the same initial conditions for each run, the final deployed networks were very different in both positioning for each

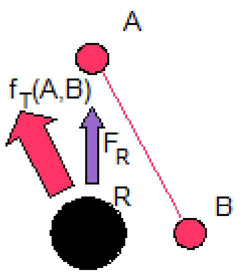

Fig. 1. Tangent force, $\mathbf{f}_T$, purple arrow, determined by two point masses **A** and **B**.

robot, and frequently in the overall pattern of dispersion, due to unpredictable multithreading effects. The overall coverage and connectivity did not drastically change from run to run, but the networks in existence in the end were qualitatively different. Another issue is that robots can become stuck in suboptimal positions, causing the entire system to also become stuck in a suboptimal configuration. Less frequent, but still possible, is the occurrence of periodic oscillations in robot actions. Finally, a small fluctuation in one robot's position can cause other robots to adjust their own positions and this can ripple through the entire network. We found that incorporating a *dead-zone* in which a robot is less or non-responsive to changes limits these effects since minor changes in a neighbor's position will not necessarily force a reassessment, which in turn damps the effects of minor perturbations on the network as a whole.

3.2 Counting Hops with Alarms

There are two phases in our algorithm. In the first phase the robots select a common reference point, or leader, and in the second phase they use the leader to coordinate movement. We assume that each robot has a unique ID and that the robot with ID 1 has been elected as the leader. To select a leader a $O(r)$ algorithm is sufficient to find the robot with the lowest ID, where r is the number of robots. This is achieved by each robot broadcasting the minimum of its own ID and the lowest ID that it has ever received. After r iterations, the entire (connected) network has the same ID as the lowest, which becomes the leader.

Once the leader is established, robots count hops to the leader. A neighbor hop count of zero is assigned to the leader itself. Other robots calculate their own hop count by measuring broadcasts from neighbors and assigning themselves a hop count corresponding to the lowest neighbor hop count plus one, as in equation 5, where H_R is the hop count of robot R, and N_R are its neighbors:

$$H_R = \min_{N \in N_R} (H_N) + 1 \tag{5}$$

Figure 2 demonstrates how counting hops works. While moving, each robot should maintain at least one neighbor with a lower hop count. The leader does

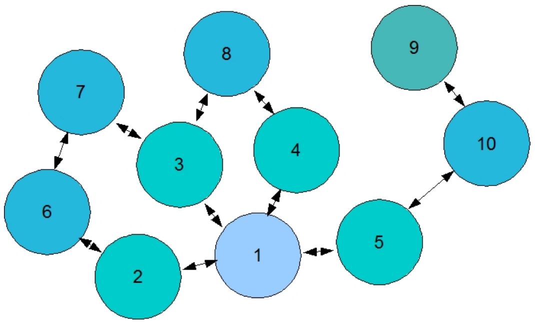

Fig. 2. Counting hops. The leader is Robot 1. Robots 2, 3, 4, and 5 are one hop away. Robots 6, 7, 8, and 10 are two hops away. Robot 9 is three hops away.

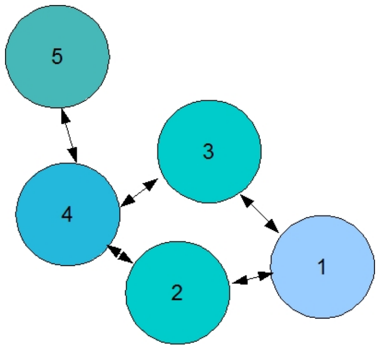

Fig. 3. Robot 5 stays put since it has only Robot 4 as a neighbor with a lower hop count. Robot 4 has two neighbors (2 and 3) with a smaller hop count so it may move and disconnect Robot 5.

not move and provides an anchor for the entire network. This reduces disconnections in the network, but it is still possible for robots to drift apart, severing the network, as shown in Figure 3. To avoid splitting the network, we add alarm messages. A robot sends an alarm if and only if there are no robots with a lower hop count within a given threshold of its communication range. We use .8 of the maximum communication range as the threshold. An alarm has a target – the target must immediately stop and freeze. This is the algorithm each robot uses:

Algorithm 1 Alarm Counting Hops – Robot State Decision Flow

```
1: procedure RobotDecisionLoop
2:     if robot has $\geq$ 1 neighbor broadcasting an alarm signal then
3:         State $\leftarrow$ Freeze
4:     else if State == Freeze and robot has $>$ 1 neighbor with lower hops then
5:         State $\leftarrow$ Scatter
6:     else if robot has $\leq$ 1 neighbor with a lower hop count then
7:         robot sends alarm to neighbor with lowest hop count, then ID.
8:         State $\leftarrow$ Freeze
9:     else if robot is receiving an alarm then
10:        State $\leftarrow$ Freeze
11:    else if robot is stuck then
12:        20% chance State $\leftarrow$ Backup
13:        20% chance State $\leftarrow$ RandomTurn
14:        20% chance State $\leftarrow$ Forward
15:        20% chance State $\leftarrow$ Random
16:        20% chance State $\leftarrow$ Scatter
17:    end if
18: end procedure
```

The robot behaviors are:

– *Random:* robots pick a random turn velocity and forward velocity.
– *RandomTurn:* robots reorient themselves in place for 5 seconds picking a random direction to rotate.
– *Scatter:* robots follow the forces in the artificial potential field.
– *Forward:* identical to *Scatter*, except the resultant vector adds a strong positive vector in the current robot direction. This has the effect of influencing the robot to continue moving in the direction it is facing.
– *Backup:* also identical to *Scatter*, except the resultant vector adds a strong vector in the opposite direction of the current robot direction.
– *Freeze:* is triggered when either a robot is sending an alarm, or another robot is sending an alarm to it.

The initial behavior for all robots is *Scatter*. When a collision occurs or a robot gets stuck, the robot randomly picks another behavior, pursues it for a few seconds, and keeps on picking a new behavior until it is once again in *Scatter*. The exception is *Freeze* which instructs the robot to cease moving. The leader stays in *Freeze* and does not move, thereby providing an anchor for the network.

4 Experiments

We utilized the Player/Stage [11] robot simulation environment for our experiments using virtual Pioneer robots. Robotic controllers were implemented in python, using the SWIG python interface to the native C Player/Stage libraries. We assume that the intensity of the wireless signal is proportional to the inverse square of distance, with uniform degradation of signal. We also assume robots can identify the source of a signal by ID, and that a robot's signal cannot be perceived outside of its communication radius.

In order to test the generalizability of our approach, we used four different worlds: cave, autolab, hospital, and house. The first three are distributed with the Player/Stage project. The last is a custom drawing of a house. The environments vary in complexity with the cave and autolab environments being the easiest, the house environment being slightly more difficult, and the hospital being the most difficult due to the many rooms. The starting locations of the robots also have an effect on the complexity of the dispersion problem— some arrangements are easier to disperse from effectively than others. The robots were placed in a tightly packed area and in such a way that the network is connected. The connectedness is a requirement for this algorithm to work, as it does not inherently contain any method for repair, except that a robot can reconnect via random wandering.

4.1 Theoretical Maximum Coverage

Figure 4 shows the minimum overlap of two robots which are in direct communication range. The edges of each of their wireless ranges, depicted by the circles, must include the other robot, which corresponds to the center of the

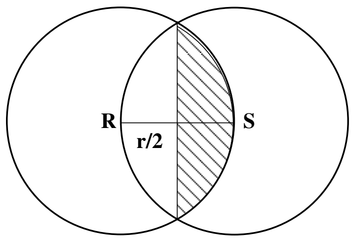

Fig. 4. Robots R and S in communication range

other circle. Now assume that each robot has a wireless range with a radius of r meters (in our experiments we use $r = 2$). The minimum region of overlap between two circles is then twice the shaded area $2(\frac{1}{3}r^2\pi - \frac{\sqrt{3}}{4}r^2) = (\frac{2}{3}r^2\pi - \frac{\sqrt{3}}{2}r^2)$ For n robots in a connected network, there must be at least $n - 1$ robots mutually in connection range (just as when a connected graph has V nodes, there must be at least V-1 edges). Thus, the minimum possible overlap is necessarily $(n-1)(\frac{2}{3}r^2\pi - \frac{\sqrt{3}}{2}r^2)$. It follows then the maximum possible coverage for n robots is $n\pi r^2 - (n-1)(\frac{2}{3}r^2\pi - \frac{\sqrt{3}}{2}r^2)$. Note that this network corresponds to a linear chain of robots. While this maximizes coverage, it is not necessarily feasible in any given environments, nor is it even desirable. One severe drawback is the brittleness of this network – the loss of any robot besides the two on the ends would fracture the network into two pieces. Nonetheless, it serves as an upper bound on the achievable coverage.

4.2 Computing Bounds with Incremental Greedy Algorithm

Since the theoretical upper bound is too lose and not achievable, we implemented an incremental greedy algorithm, similar to the one in [6] but ignoring obstacles.

Algorithm 2 places sequentially each robot to maximize the total network coverage while keeping connectivity. Robots may be placed at any legal location within wireless range, including on the far side of walls or other obstacles. The quantity maximized at each step is the total visual area of the robot swarm.

Note that the results depend on the sampling distance of points to produce **C**. We experimented with different values, from every 32 cm down to 8 cm. The spatial configurations of the networks obtained, shown in Figure 5, are noticeably different, despite the same initial conditions. Figure 6 shows the results in a more complex environment. Presumably, continuing to increase the sample rate would yield limiting behavior, but since the run time is quadratic in the sampling rate, this limit is expensive to find. Also since the algorithm is greedy, choosing an initially profitable location might ultimately lead to a suboptimal placement. With a sampling rate of 8 cm and with optimizations to reduce repeated sampling it took about 8 hours on a 3.6GHz computer to deploy 24 robots. Once again, our goal is to find an upper bound and provide a metric of reference.

Algorithm 2 Incremental greedy algorithm

```
1:  procedure IncGreedy($\mathbf{R}, x_0, y_0$)
2:      $\mathbf{R}(1) \leftarrow (x_0, y_0)$;                                    ▷ position of first robot
3:      $\mathbf{C} \leftarrow$ Visual Coverage of R(1); $\mathbf{W} \leftarrow$ Wireless Coverage of R(1); $r \leftarrow 2$
4:      while $r <= |\mathbf{R}|$ do                                              ▷ for each robot
5:          $\mathbf{C}_{best} \leftarrow C$; Area$_{best} \leftarrow 0$
6:          for $w = (w_x, w_y) \in \mathbf{W}$ do
7:              $\mathbf{C}_{cur} \leftarrow$ Visual Coverage of R(1), ... ,R(r); $Area \leftarrow sum(C_{cur})$
8:              if $Area > Area_{best}$ then
9:                  $\mathbf{C}_{best} \leftarrow C_{cur}$; $\mathbf{R}_{best} \leftarrow (w_x, w_y)$
10:                 $\mathbf{W} \leftarrow$ Wireless Coverage of R(1), ..., R(r)
11:             end if
12:         end for
13:         $\mathbf{R}(r) \leftarrow \mathbf{R}_{best}$; $\mathbf{C} \leftarrow \mathbf{C}_{best}$; $r \leftarrow r + 1$
14:     end while
15:     return $\mathbf{R}, \mathbf{C}$                    ▷ robot positions are in $\mathbf{R}$ and coverage is in $\mathbf{C}$
16: end procedure
```

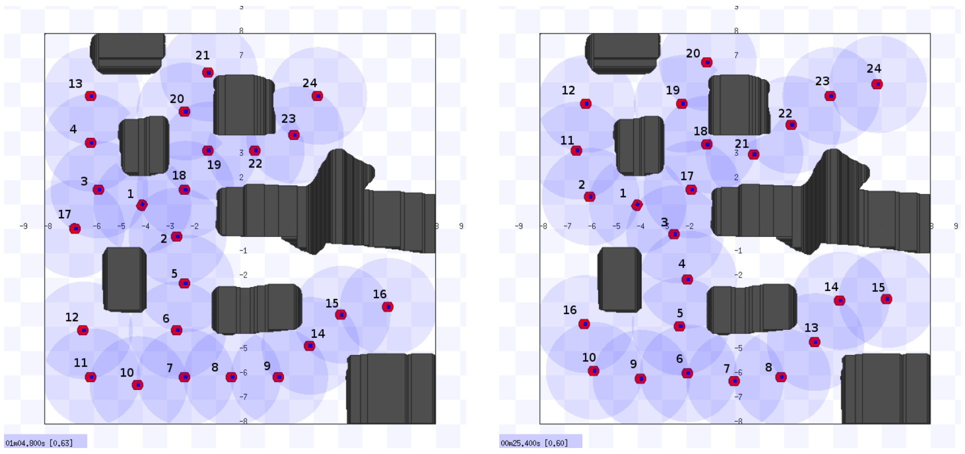

Fig. 5. Cave– Incremental greedy algorithm for different sample distances. (left) 32 cm, (right) 8 cm. The numbers show the order in which the network was deployed.

Figure 7 shows the coverage obtained after the placement of each successive robot. Note that although the values are not directly comparable due to differences in the initial robot placement and the environment, the more constricted and complex the environment, the lower is the final coverage obtained. Note that as the complexity of the environment increases, the solution obtained by the incremental greedy algorithm becomes unrealistic. This is because the incremental greedy algorithm skips over walls and other barriers, which makes it impossible in reality to replicate the coverage results obtained by this method. Again though, we use this as a a bound for the coverage obtainable.

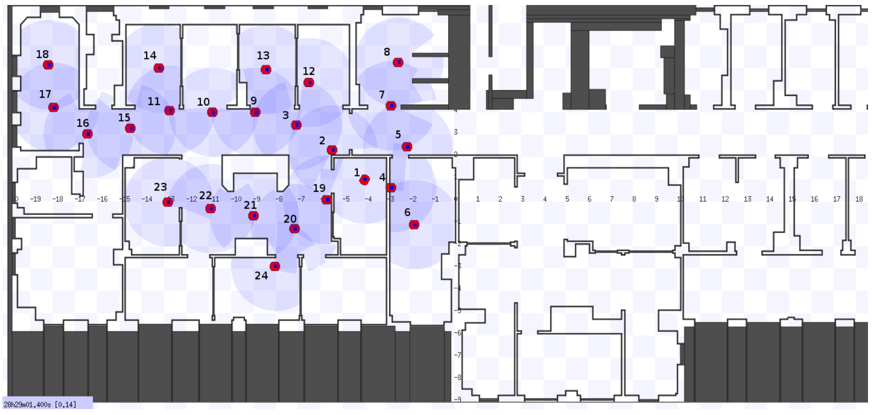

Fig. 6. Hospital Section– Incremental greedy algorithm, .08 sample distance

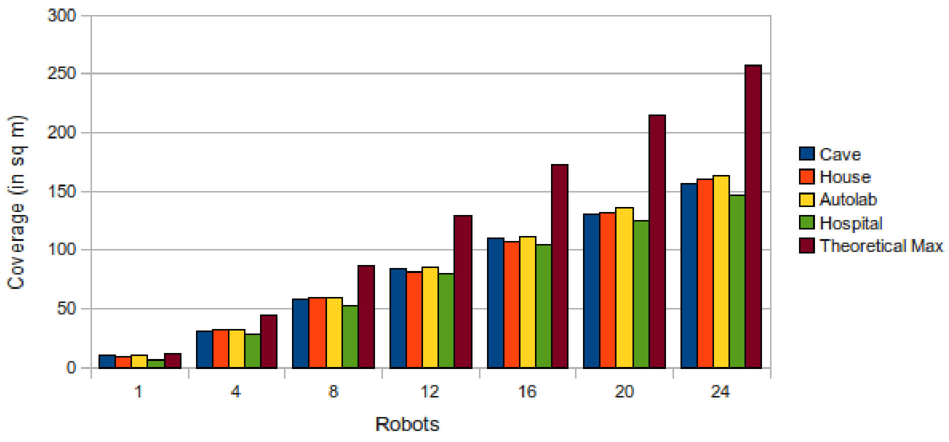

Fig. 7. Coverage of incremental greedy algorithm for increasing numbers of robots in different environments

4.3 Results on the Counting Hops with Alarms Algorithm

We performed three set of experiments of the Hop Counting with Alarms algorithm. The first was to measure sensitivity caused by minor changes in robot position, the second to measure performance on different environments, and the last to measure the effects of reducing the number of laser readings.

1. Sensitivity analysis.. To measure sensitivity to initial positions we compared the results of 20 runs with fixed initial positions against 20 runs with randomized initial positions of 24 robots in the cave environment. The fixed positions coverage increases from a starting coverage of $35.372m^2$ to an average of $81.443m^2$, After 5 minutes, the robots covered $74.002m^2$, and 10 minutes, $78.182m^2$. Thus roughly 83.85% of the expansion is done by 5 minutes, and roughly 92.92 by 10 minutes. The final total fell well short of the maximum

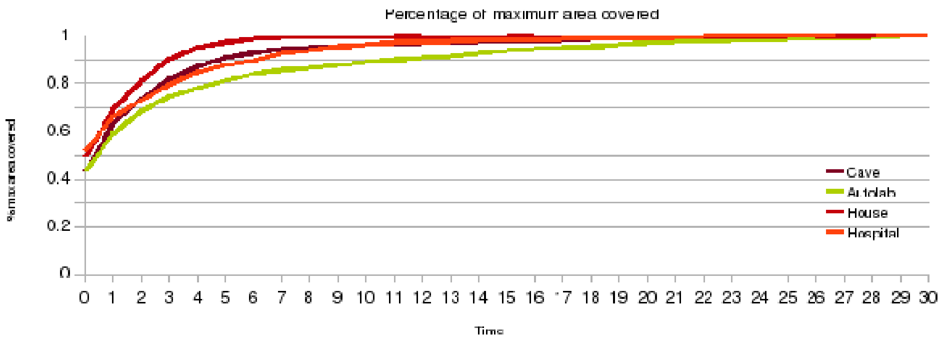

Fig. 8. Coverage over time in different environments as percentage of the coverage achieved in 30 minutes (average of 20 runs).

coverage predicted by the incremental greedy algorithm, obtaining $81.433m^2$ / $156.5972m^2$ in the end, or 52.00% of the possible expansion. Again though, the incremental algorithm is not actually possible in a real deployment.

With random placement the coverage increases from a starting coverage of $35.372m^2$ to an average of $82.244m^2$. After 5 minutes, the average coverage was $74.775m^2$, and 10 minutes, $79.05m^2$. Thus 84.07% of the expansion is done by 5 minutes, and by 10 minutes roughly 93.23% of the final coverage is completed. There was little or no difference between the fixed and randomized cases, which made us to conclude that at least in terms of aggregate behavior, the algorithm was not sensitive to minor positional variations.

2. Performance in different environments. The second set of experiments included twenty runs of each of the autolab, house, and hospital environments mentioned earlier. The cave and autolab environments were the more forgiving, while the house and especially the hospital proved more difficult because robots had to leave or enter rooms to maximize coverage. To emphasize the progress with time, the results in Figure 8 show the percentage of the coverage achieved after 30 minutes, since at that point the coverage had plateaued.

In the house environment coverage increased from $37.792m^2$ at the start to $73.995m^2$ after 5 minutes. $75.867m^2$ after 10 minutes, and $76.128m^2$ after 30 mninutes. Thus, by 5 minutes, 94.44% of the expansion was complete; by 10 minutes, 99.3% was finished. In this environment, the robots spread out almost entirely in the original room, and ignored the adjoining room. The final coverage of $76.128m^2$ was also short of the $160.27m^2$ obtained by the incremental greedy algorithm. In the autolab environment, the final dispersion achieved a coverage of roughly $99.343m^2$. After 5 minutes, the dispersion was $80.791m^2$, which represented 66.96% of the total dispersion; after 10 minutes, the total dispersion was roughly $88.564m^2$ which was 80.98% of the total dispersion. The hospital environment highlighted the inadequacies of the algorithm in handling narrow doorways. The robots in the hallway did manage to spread out, but robots in the room for the most part never managed to escape the room. Frequently, the doorway would become blocked during the course of a run, and that would prevent robots from leaving. The final expansion was on average around 59.205m^2. after 30 minutes, starting from an initial coverage of 30.833. After 5 minutes, the algorithm covered 51.95m^2, representing 74.43% of the total expansion. After 10 minutes, the average coverage was 57.008m^2, which represented an average of 92.26% of the total expansion achieved by the algorithm. The total expansion that the algorithm achieved, 59.205m^2 was also well short of the maximum 146.759m^2 that the incremental greedy algorithm obtained in this environment, in large part because of the inability to get many robots out of the starting room.

The algorithm had similar characteristics across all environments. Most of the coverage was obtained in the first 5 to 10 minutes, with very small gains in further coverage after that point. The algorithm did not do well in complex environments with doorways and enclosed rooms– the hospital environment in particular was quite difficult. As mentioned before, robots already in the hallway fared well, while robots in the room were unsuccessful in leaving the room. Overall the ratio of coverage of our algorithm to the increemental greedy algorithm is 0.53 for the cave, 0.61 for the autolab, 0.48 for the house, and 0.4 for the hospital.

3. Dispersion with fewer laser measurements. The last set of experiments was with reduced numbers of laser readings on each robot. We tried several different conditions, from the 360 measurements we used in earlier experiments down to 180, 90, 45, 20, and 6. In each case the measurements are equidistant. The algorithm is fairly robust to a reduction in resolution; only when the number of laser readings was reduced to 6 was there a significant decline in observed coverage. However, even in that case, the robots were able to disperse to cover 183.42% of their initial coverage indicating that even very limited robots may succeed at dispersing, although not as much as with denser sensor readings.

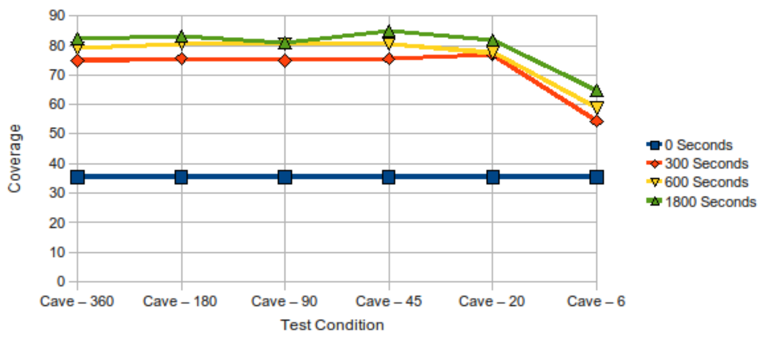

Fig. 9. Coverage with fewer laser range-finder readings in the cave environment

5 Conclusions and Future Work

We have proposed a new algorithm for robotic dispersion which uses leader-election, counting hops, and potential fields to disperse robots and create a mobile sensor network. We have demonstrated within Player/Stage that this approach achieves a significant coverage, while maintaining connectivity, requiring robots with only wireless communication and a laser range-finder. We have also shown that the approach is robust to changes in initial conditions, that it can be applied effectively to robots with range-finders with significantly reduced resolution, and that it works in different types of environments.

There are new and exciting directions for future research. A systematic approach to set values of parameters, such as that provided by a learning algorithm or an evolutionary approach, would likely improve the performance across environments. Another extension is to incorporate visual data to enable robots to move more intelligently. Currently robots do not distinguish robots in their visual field from other obstacles. This would help deal with some of the issues in narrow openings and help robots to spread out in a more coordinated fashion. Finally we would like to implement this approach with real robots.

References

1. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Rob. Res. **5**(1) (1986) 90–98
2. Morlok, R., Gini, M.: Dispersing robots in an unknown environment. In: Proc. Int'l Symp. on Distributed Autonomous Robotic Systems. (2004)
3. Brooks, R.A.: A robust layered control system for a mobile robot. Journal of Robotics and Automation **RA-2**(1) (March 1986) 14–23
4. Howard, A., Matarić, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Proc. Int'l Symp. on Distributed Autonomous Robotic Systems. (2002) 299–308
5. Poduri, S., Pattem, S., Krishnamachari, B., Sukhatme, G.S.: Using local geometry for tunable topology control in sensor networks. IEEE Transactions on Mobile Computing **8**(2) (2009) 218–230
6. Howard, A., Matarić, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots **13**(2) (2002) 113–126
7. Batalin, M., Sukhatme, G.S.: The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. IEEE Transactions on Robotics **23**(4) (Aug 2007) 661–675
8. Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. Autonomous Robots **11**(3) (2001) 319–324
9. McLurkin, J., Smith, J.: Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In: Proc. Int'l Symp. on Distributed Autonomous Robotic Systems. (2004)
10. Ludwig, L., Gini, M.: Robotic swarm dispersion using wireless intensity signals. In: Proc. Int'l Symp. on Distributed Autonomous Robotic Systems. (2006) 135–144
11. Gerkey, B., Vaughan, R., Howard, A.: The Player/Stage project: Tools for multi-robot and distributed sensor systems. In: 11th Int'l Conf. on Advanced Robotics. (2003)